СБЕР

Auto DS Agent (Code Actor) - для задач принятия решений

DS Agent - Code Actor

Я сделал выгрузку по сотрудникам за прошлые года по их метрикам продуктивности и другим показателям. Так же в этих данных есть история их повышений. Построй модель, которая будет определять стоит ли повысить сотрудника.

department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service	KPIs_met >80%	awards_won?	avg_training_score	target
Sales & Marketing	region_7	Master's & above	f	sourcing	1	35	5.0	8	1	0	49	0
Operations	region_22	Bachelor's	m	other	1	30	5.0	4	0	0	60	0
Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34	3.0	7	0	0	50	0
Sales & Marketing	region_23	Bachelor's	m	other	2	39	1.0	10	0	0	50	0
Technology	region_26	Bachelor's	m	other	1	45	3.0	2	0	0	73	0
...	...	...	...	...	...	...	...	...	...	...	...	...
Technology	region_14	Bachelor's	m	sourcing	1	48	3.0	17	0	0	78	0
Operations	region_27	Master's & above	f	other	1	37	2.0	6	0	0	56	0
Analytics	region_1	Bachelor's	m	other	1	27	5.0	3	1	0	79	0
Sales & Marketing	region_9	NaN	m	sourcing	1	29	1.0	2	0	0	45	0
HR	region_22	Bachelor's	m	other	1	27	1.0	5	0	0	49	0

GIGA CHAT

scikit learn

LAMA

CatBoost

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, roc_curve

# 1. Загрузка данных
df = pd.read_csv("10101_binary_class_test.csv", index_col=0)

# 2. Подготовка данных
X = df.drop(columns=['target'])
y = df['target']

# 3. Разделение данных
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# 4. Обучение модели
model = LogisticRegression()
model.fit(X_train, y_train)

# 5. Предсказание вероятностей
y_pred_proba = model.predict_proba(X_test)[:, 1]

# 6. Расчет ROC AUC
roc_auc = roc_auc_score(y_test, y_pred_proba)
print(f'ROC AUC: {roc_auc}')
```

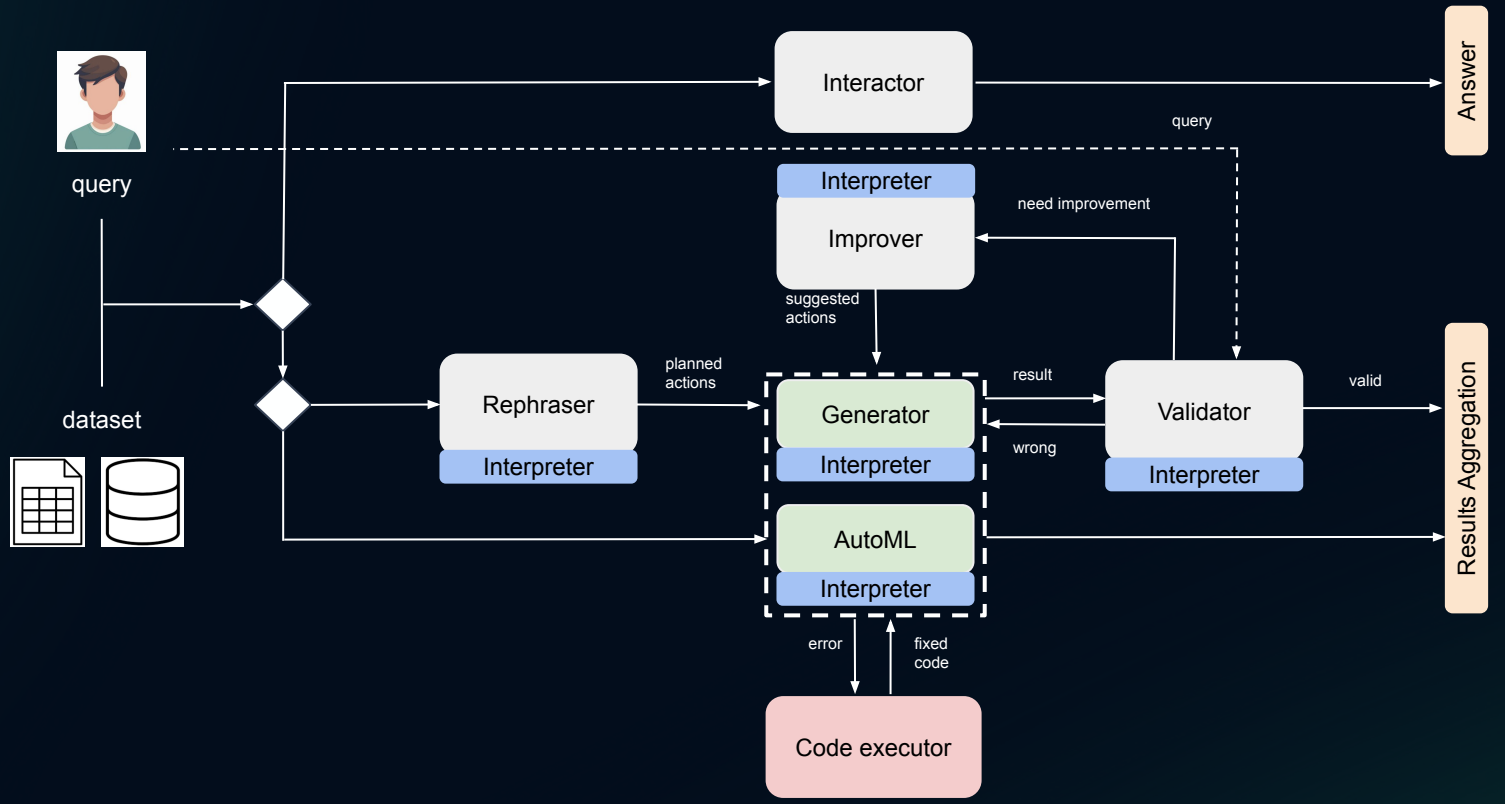
code_improvement_agent: В результате выполнения кода мы оцениваем производительность модели логистической регрессии с использованием метрики ROC AUC для бинарной классификации. Если итоговое значение ROC AUC низкое (например, меньше 0.7), это может указывать на то, что модель не смогла эффективно различать классы.

Объяснение возможных причин низкого результата:

1. **Неоптимальные признаки**: Признаки, которые были выбраны для обучения, могут не содержать достаточной информации для эффективного обучения модели. Например, если некоторые признаки имеют низкую значимость или если их значения сильно коррелируют, это может привести к снижению производительности.
2. **Смещение по классам**: Если классы в целевой переменной (например, 0 и 1) сильно несбалансированы, это также может негативно сказаться на качестве модели.

Код и обученная модель с высококачественными метриками

DS Agent - архитектура



Rephraser:

Для заданного запроса он планирует действие по эффективному написанию кода

Generator:
Создать код в соответствии с запланированными и предложенными действиями

AutoML:
Запустить AutoML Instruments

Validator:

Проверяет результаты для заданного запроса и решает, необходимо ли и возможно ли улучшение метрики

Improver:

Анализ предыдущего кода и результатов, а также предложение новых способов улучшения обучения модели

Interactor:

Взаимодействуйте и общайтесь с DS Agent

Interpreter:

Переводит техническое описание в понятный человеку текст

DS Agent - результаты

DS Агент обходит агентов SOTA и DS-ов при выполнении типичных задач Kaggle

Dataset	k25	k50	k75	AutoKaggle	Aide	DS Agent
Titanic	0.76555	0.77511	0.77751	0.78	0.71	0.78
Space Titanic	0.77554	0.79214	0.80009	0.77	0.79	0.797
House Prices	0.856869	0.873576	0.884658	0.862	0.883	0.885
Monsters	0.72589	0.73913	0.74669	0.723	0.721	0.77
Academic Success	0.82788	0.83418	0.83626	0.82	0.835	0.835
Bank Churm	0.87075	0.8879	0.8912	0.86	0.79	0.88
Obesity Risk	0.89279	0.90308	0.90688	0.896	0.896	0.90
Plate Defect	0.85195	0.8793	0.88564	0.82	0	0.88

DS Agent - примеры

Задача:

Возьми датасет, сначала построй кластеризацию в ансупервайзд режиме, после чего для каждого кластера построй модель логистической регресии. Используй самые простые алгоритмы

Решение:

```python
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler

# Шаг 1: загрузка данных
df = pd.read_csv('datasets/breast_cancer_train.csv')
print("Columns:", df.columns)
print("\nFirst rows of the dataset:")
print(df.head())

# Шаг 2: предварительная подготовка данных
X = df.drop(columns=['target'])
y = df['target']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Шаг 3: построение кластеризации (K-means)
kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

# Добавляем результаты кластеризации обратно в DataFrame
df['cluster'] = clusters

# Шаг 4: создание отдельных моделей логистической регрессии для каждого кластера
for cluster in range(2): # перебираем оба кластера
    subset = df[df['cluster'] == cluster]

    X_subset = subset.drop(columns=['target', 'cluster'])
    y_subset = subset['target']

    # Делим набор данных на тренировочный и тестовый
    X_train, X_test, y_train, y_test = train_test_split(
        X_subset, y_subset, test_size=0.2, random_state=42
    )

    # Создаем и тренируем модель логистической регрессии
    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)

    # Прогнозирование и оценка качества модели
    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    report = classification_report(y_test, predictions)

    print(f"\nCluster {cluster} Accuracy: {accuracy:.4f}")
    print(report)
```

Код обучения модели

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from joblib import dump

# Загрузка обучающего набора данных
df = pd.read_csv('datasets/breast_cancer_train.csv')

# Предварительная обработка данных
X = df.drop(columns=['target'])
y = df['target']

# Масштабирование признаков
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Кластеризация методом K-means
kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

# Добавление результатов кластеризации обратно в dataframe
df['cluster'] = clusters

# Обучение отдельных моделей логистической регрессии для каждого кластера
models = {}
for cluster in range(2):
    subset = df[df['cluster'] == cluster]
    X_subset = subset.drop(columns=['target', 'cluster'])
    y_subset = subset['target']

    # Разделение на тренировочную и проверочную выборки
    X_train, _, y_train, _ = train_test_split(
        X_subset, y_subset, test_size=0.2, random_state=42
    )

    # Создание и обучение модели логистической регрессии
    model = LogisticRegression(max_iter=1000)
    model.fit(X_train, y_train)

    models[f'cluster_{cluster}_model'] = model

# Сохраняем масштабатор и все модели
dump(scaler, 'code/scaler.joblib')
dump(models, 'code/models.joblib')
dump(kmeans, 'code/kmeans.joblib')  # добавляем сохранение объекта KMeans
```

Код для инференса

```python
import pandas as pd
from joblib import load

# Загружаем ранее сохраненные модели и масштабатор
scaler = load('code/scaler.joblib')
models = load('code/models.joblib')
kmeans = load('code/kmeans.joblib')  # загружаем объект KMeans

# Загрузка тестовых данных
test_df = pd.read_csv('datasets/breast_cancer_test.csv')

# Проверка наличия столбца 'id' перед попыткой удаления
if 'id' in test_df.columns:
    # Выделение признаков и целей из тестового набора
    X_test = test_df.drop(columns=['id'])
else:
    X_test = test_df.copy()

ids = test_df.get('id', None)  # получаем идентификаторы если они есть

# Удаляем целевой признак "target" из тестовых данных
X_test = X_test.drop(columns=['target'], errors='ignore')

# Масштабирование признаков
X_test_scaled = scaler.transform(X_test)

# Предсказываем кластеры для тестовой выборки
predicted_clusters = kmeans.predict(X_test_scaled)  # используем уже обученный kme

# Применение соответствующих моделей по каждому кластеру
final_predictions = []
for i in range(len(predicted_clusters)):
    cluster_id = predicted_clusters[i]
    model_name = f'cluster_{cluster_id}_model'
    prediction = models[model_name].predict([X_test.iloc[i]])
    final_predictions.append(prediction[0])

# Формирование результата в виде DataFrame
result_df = pd.DataFrame({'id': ids, 'prediction': final_predictions})

# Сохранение прогнозов в CSV-файл
result_df.to_csv('breast_cancer_test_predictions.csv', index=False)
```